The GLOBE Program and Software Based Telemetry Processing

Steve Duran, Chris Wilkinson, Michael Robbins,
Karen Michael, Greg Henegar

Data Processing Systems Branch, Code 514
Mission Operations and Data Systems Directorate
NASA, Goddard Space Flight Center
Greenbelt, Maryland 20771

Abstract:

A low cost approach to processing telemetry data is described. Telemetry processing functions such as frame synchronization and Reed-Solomon error control, traditionally performed in custom hardware circuits, were developed in software and run on low cost desktop Unix workstations and portable laptop computers. This technology was successfully applied in support of the Global Learning and Observations to Benefit the Environment (GLOBE) program, and has definite future application in the development of very low cost ground data systems.

The GLOBE Program

The National Aeronautics and Space Administration (NASA) is participating in the GLOBE program by providing satellite data relay service to certain schools. The system provides a means for students to transmit their environmental measurements to the National Oceanic and Atmospheric Administration (NOAA) and receive processed data back. NASA provides telemetry formatting and processing software such as frame synchronization and Reed-Solomon error control, use of the Tracking Data and Relay Satellite System (TDRSS) Multiple Access (MA) services, and Student Communication Unit (SCU) transmitters and receivers. Figure 1 depicts the various elements of the GLOBE program and the key data flows.

A typical data flow scenario starts with the student taking field measurements(e.g. temperatures, wind speeds). These measurements are formatted into a data file and transmitted at a pre-determined time using the SCU transmitter. The transmission is relayed by the TDRSS satellite to the ground terminal at White Sands, New Mexico where the data is fed into the NASA Communications (NASCOM) network and transmitted back to the Goddard Space Flight Center (GSFC) in Greenbelt, Maryland. At GSFC, the data is received by the GLOBE Data Processing System, which performs software based telemetry processing on the data stream, and reconstructs the original data file transmitted by the student. The reconstructed file is then transmitted via the Internet from GSFC to NOAA's Forecast Services Laboratory (FSL), where data analysis and processing is performed. Results from the NOAA processing are then transmitted back to the student at a later time, using the reverse path of the original "return link". The student receives the results on the SCU receiver.

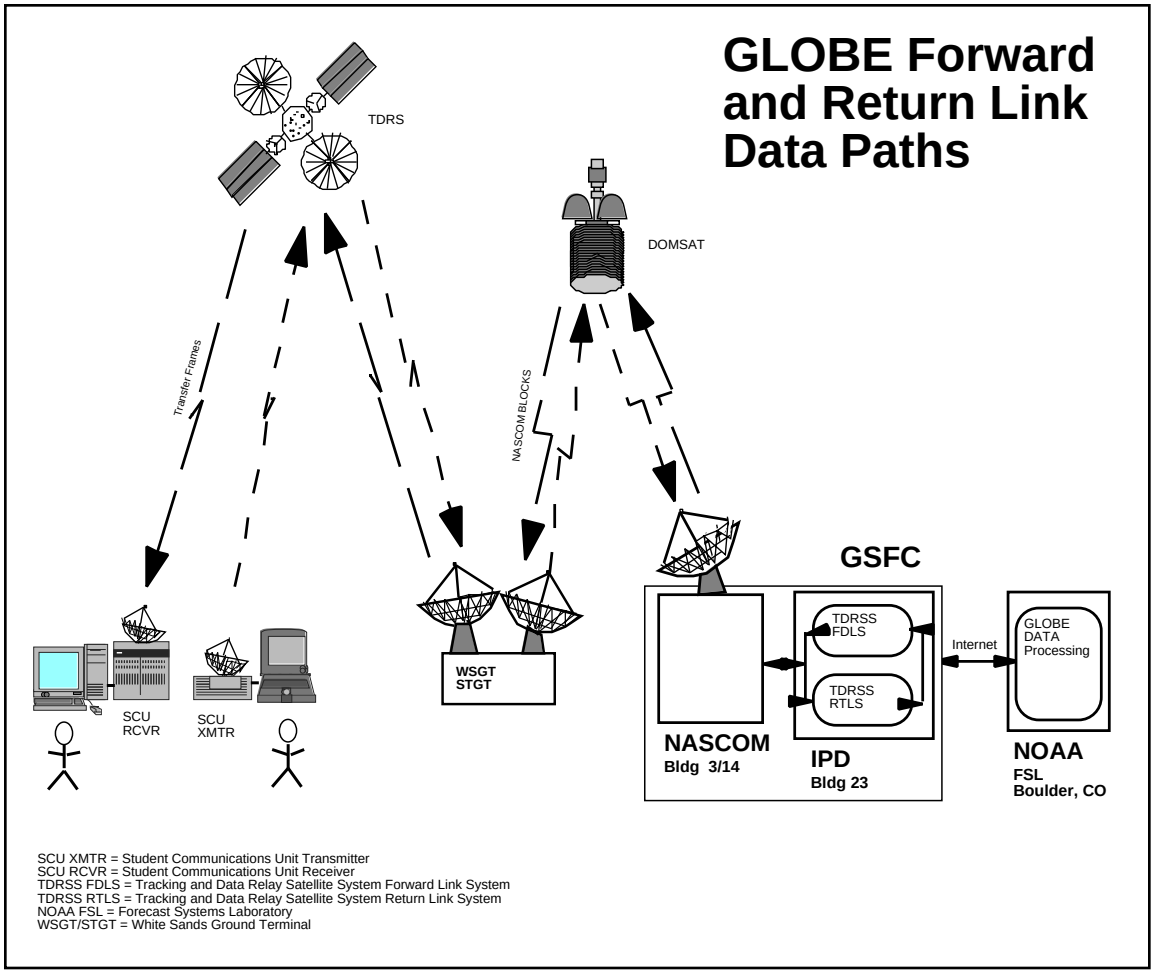

Figure 1: GLOBE Forward and Return Link Data Paths

Software Based Telemetry Processing

The Data Processing Systems Branch at NASA's Goddard Space Flight Center has traditionally designed and operated ground based systems for processing the return link telemetry from earth orbiting satellites. Many of these satellites relay their data via NASA's Tracking and Data Relay Satellite System. The "satellite" in this instance is the student's computer. Telemetry formatting and telemetry processing was required for both sides of the forward and return link paths.

Historically, the least expensive method for performing telemetry processing has been custom digital logic circuits. The processes are bit manipulation intensive. Previous trade studies comparing hardware to software showed the cost of the computer to be the key factor. The evolution of commercial computer technology has produced low cost and high performance computers. This technology is a cost effective host for software based telemetry processing.

Two common "building blocks" in NASA's ground data systems are the frame synchronizer and the forward error correction functions. NASA adopted a set of space data standards in the 1980's known as the Consultative Committee for Space Data Standards (CCSDS) Blue Books. These specifications set standards for both frame synchronization and forward error corrections. The Blue Books specify frame synchronization patterns and frame lengths, and defines a

Reed-Solomon code for reliable forward error correction. The following sections describe these telemetry processing functions in more detail, as well as how these functions were adapted to a cost effective software based approach.

Telemetry frame synchronization and forward error correction functions have been implemented in Application Specific Integrated Circuits (ASIC's) and integrated into a tightly coupled real-time environment. This would typically provide a high performance level (e.g.10 Megabits per second) at a given fixed cost. Thus low and medium data rate applications had to absorb the same cost as high rate applications. The evolution of commercial computer technology has produced low cost, high performance off-the shelf computers, suitable for use as host platforms for software based telemetry processors.

Software based telemetry processing allows the data system engineers to match the performance requirements of a data system with a scalable telemetry processing function hosted on an appropriate Commercial off-the shelf (COTS) PC or Unix workstation. This results in cost savings for most applications, which typically operate at low or medium data rates (below 2 Megabits per second).

In order to lower development, replication, and maintenance costs, several telemetry processing functions for the GLOBE data processing systems were performed in software, rather than hardware. The functions are telemetry formatting, Reed-Solomon encoding and decoding, frame sychronization, convolutional encoding, and both frame and NASCOM block level cylic redundancy check (CRC). Also, a simple packet procesor was developed to process GLOBE data. Each of these software modules have been used as building blocks in the telemetry processing systems for various missions. The most noteworthy of these functions are the Reed-Solomon decoder and the frame synchronizer which will be described in detail in the following two sections.

Software for the Frame Synchronization Process

The frame synchronization process involves a bit by bit search of the received data to locate the synchronization pattern. The synchronized data is then aligned to byte boundaries for subsequent processing. These first two steps require extensive bit manipulation and pose a performance challenge to a software approach.

The frame synchronization software was written to be flexible in order to handle multi-missions, therefore the user must enter set up parameters in order to customize the software to the application. The software allows the user to enter such parameters as the frame synchronization pattern, the length of the frame, and the error tolerance.

The frame synchronization process employs a set of strategies that ensure that false acquisition of data is minimized. These strategies are embodied in a state machine containing four states, or "modes of operation" which set the rules for the synchronization process. The four Modes of operation for frame synchronization are: Search, Check, Lock, and Flywheel. (See Figure 2) Search Mode can be re-entered from any of the other 3 modes. Upon Acquisition of Signal (AOS), Loss of Signal (LOS), or software reset the algorithm defaults to Search Mode. AOS is typified by a period of noise followed by a real signal.

Since the frame synchronization pattern can occur anywhere in this bitstream, it typically does not fall on a byte boundary. Search Mode is conducted starting with the first byte of data received and shifting left using bitwise operators in order to conduct a bit by bit search. After each shift the value contained in the register is compared against a look up table, previously generated, based on the chosen frame synchronization pattern. The use of a look up table provides many benefits including speed, programmable error thresholds, and inverted frame synchronization pattern detection.

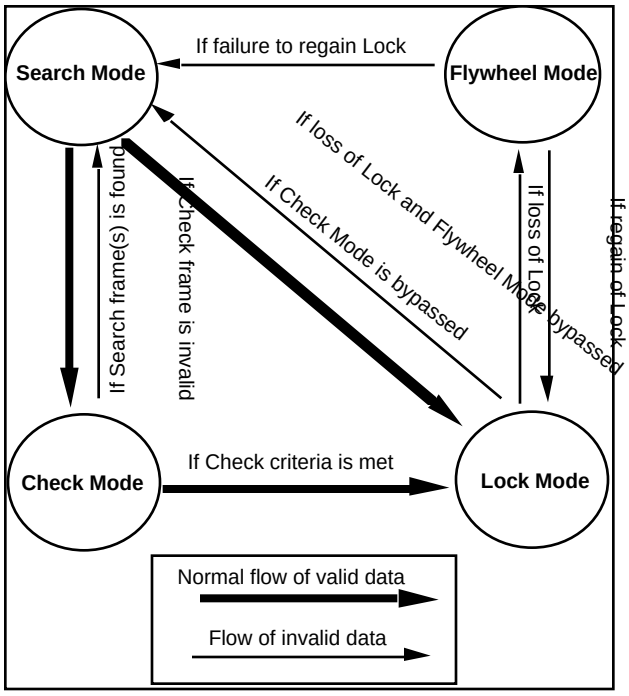

Figure 2: Frame Synchronizer State Diagram

Check, Lock, and Flywheel are not bit by bit search operations. These operations look for the synchronization pattern at the expected end of the previous frame window. The size of this window can be set by the user and can range from $^{+}/_{-}$ N bits. Occasionally the successive frame synchronization patterns will not be found at the expected locations for various reasons. For example, in some cases the bit errors exceed the error tolerance, and in other cases bit slips may occur which cause the data to be skewed left or right by more than the specified tolerance. In the latter case, if the frame synchronization pattern is not in the expected window, the frame synchronizer process enters Search Mode.

The primary mode of frame synchronization is called Lock Mode. In order to get into Lock Mode the Search and Check criteria must be passed. Entering Lock gives an indication that data is good and that the frame synchronizer should remain in Lock Mode unless the error tolerance is exceeded. If the error tolerance is exceeded the operation will "Drop Lock" indicating a Loss of Signal (LOS) and enter Flywheel Mode.

Flywheel Mode allows a programmable number of invalid synchronization patterns to be output on a premise that Lock can potentially be re-achieved at any frame. It allows for a momentary dropout or a noise hit right on the synchronization pattern. If Flywheel frames continue to exceed error tolerance and Lock cannot be re-gained then the frame synchronizer enters Search Mode. Many users prefer not to output data until Lock Mode is achieved although with software, frames can be output during any mode.

Once the frame synchronization pattern is matched the software passes the location information to the byte alignment routine. The expected end of frame is calculated according to the chosen frame length. The byte alignment routine needs to know which bit of which byte contains the start and end of the frame. Once this information is calculated, the frame is byte aligned starting with the first byte of the frame synchronization pattern (See Figure 3). The shifting is performed bitwise across the entire frame 32 or 64 bits at a time, depending on the computer's maximum register length. The frame synchronization routine runs at up to 50 Mbps on an SGI Challenge L with a 150 MHz Processor, and up to 30 Mbps on a 70 MHz SUN SPARC 5.

Frame Synchronization Pattern Slipped Left by 3 bits

HEX		D	6	7	F	E	0	E		
BIN	X000	1101	0110	0111	1111	1110	0000	1110	1XXX	

Frame Synchronization Pattern Slipped Right by 2 bits

HEX			6	B	3	F	F	0	7	
BIN		XX00	0110	1011	0011	1111	1111	0000	0111	01XX

Byte Aligned Frame Synchronization Pattern

HEX		1	A	C	F	F	C	1	D	
BIN		0001	1010	1100	1111	1111	1100	0001	1101	

Figure 3: Byte Alignment Examples

Software for the Reed-Solomon Decoding Process

An error detection and correction method is necessary since both the return and forward link data transmissions are subject to noise interference which introduces errors into the data stream. In most cases, acknowledgment/retransmit protocols are not always feasible due to long round trip delays. The CCSDS recommendation for Advanced Orbiting Systems (AOS) Grade-2 service provides error control using Reed-Solomon (RS) encoding and decoding. Current systems utilize a hardware Reed-Solomon card. The software decoding algorithm posed a design and performance challenge, due to the need for an efficient implementation of the intense calculations that are required for decoding in order to achieve usable performance.

RS coding theory is an elegant mathematical theory based on concepts from linear algebra, information theory, and finite field mathematics. The RS decoding algorithm is a five step process with a sixth step for interleaved data. Each box in Fig. 4 is one of the steps of the decoding process and corresponds to a software function. In order to realize an efficient software implementation, knowledge of the target computer architecture, explicit register use, lookup tables, and careful implementation of the decoding equations and algorithms were all used to provide an efficient software solution. The decoder is fully CCSDS compliant, capable of processing shortened RS codes and interleaved data.

Several performance evaluations were conducted on versions of the RS decoder software that had been ported to various operating system environments. The DOS executable achieved a data rate of approx. 95 kbps on a 33 MHz 486SX laptop, the UNIX executable on a 70 MHz SUN SPARC 5 had a maximum data rate of 1.1 Mbps, and an HP executable had a maximum data rate of 2 Mbps on an HP9000. These data rates are adequate for many missions.

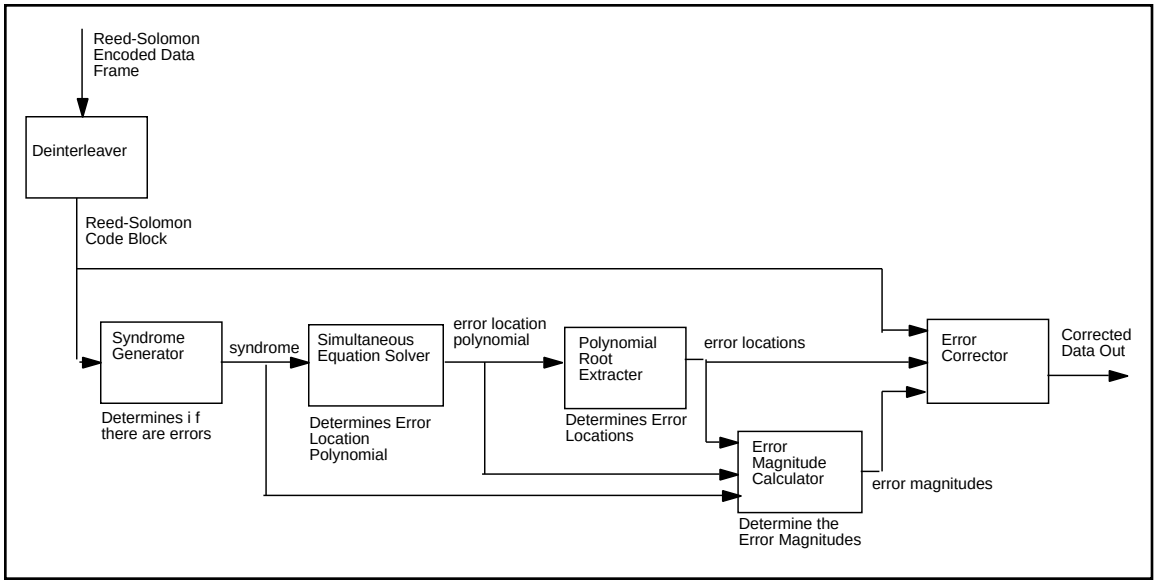

Figure 4: RS Decoding Algorithm Block Diagram

The software version of the RS decoder has several advantages over its hardware, ASIC-based counterpart. First is flexibility. With a new Galois Field symbol lookup table and new RS code parameters, the decoder could be modified to decode almost any RS code. In order to make the same modifications to a hardware decoder would mean to almost redesign the ASIC hosting the algorithm, and would be costly. Next is reliability. Software components, once proven, do not wear out or fail. The hardware platforms hosting the RS software historically have higher reliability, as well as commercial maintenance support. One area in which the software version will fall short of a hardware implementation is in high-speed performance (>10 Mbps). However, the price/performance ratios of commercial workstation should continue to improve at 30% per year. Thus the price/performance level of a software implementation will continue to improve, without the need for redesign. In the mean time, adequate performance is readily achievable for those medium and low data rate applications which comprise the bulk of upcoming spacecraft missions. Also, replication of the software decoder costs nothing, whereas replication of a hardware decoder is rather costly.

For the GLOBE project, since the forward link data was RS encoded, an RS decoder was required for the student lap top PC which operates and records the data from the portable TDRSS receiver. Rather than developing an RS decoder on a PC card, which would have been time consuming and costly, the software RS decoder that had been developed for a UNIX based machine was easily ported to the DOS/Windows PC environment.

Operations

GLOBE daily operations require very little man power. The system was designed to operate with very little operator interaction. The student transmissions are scheduled during 10 to 20 minute periods throughout the day. The operator actions take place following the school day after all the data has arrived. There are really only two routine tasks for an operator. The first one is that once data has been received and processed, an operator has to manually transfer the files over the internet to NOAA FSL. The second operator task is required when data needs to be sent on the forward link, an operator has to manually perform this operation. But with minimal effort, it

would be possible to automate both of these task, and in the near future, they may be automated.

Figure 5 below depicts the data flows between three locations for a particular GLOBE system test and demonstration. Data path 1 was the forward link data path from the Goddard Space Flight Center (GSFC) to the White Sands Ground Terminal (WSGT). From WSGT, forward link data was transmitted through paths 1A and 1B, via TDRSS to a receiver in Reston, Virginia and a receiver on the roof of the NASA headquarters building in Washington D.C. Paths 2 and 3 formed the return link data paths. Part of the test/demonstration included an engineer at the NASA headquarters site taking a picture with a digital camera, then transmitting the picture on the return link to GSFC. Once the image was received and viewed at GSFC, the image was then transmitted back on the forward link. Engineers at both receiver sites were then able to receive the image and view it. It should be noted that the real time telemetry processing, both at GSFC and at the remote sites, was performed in software.

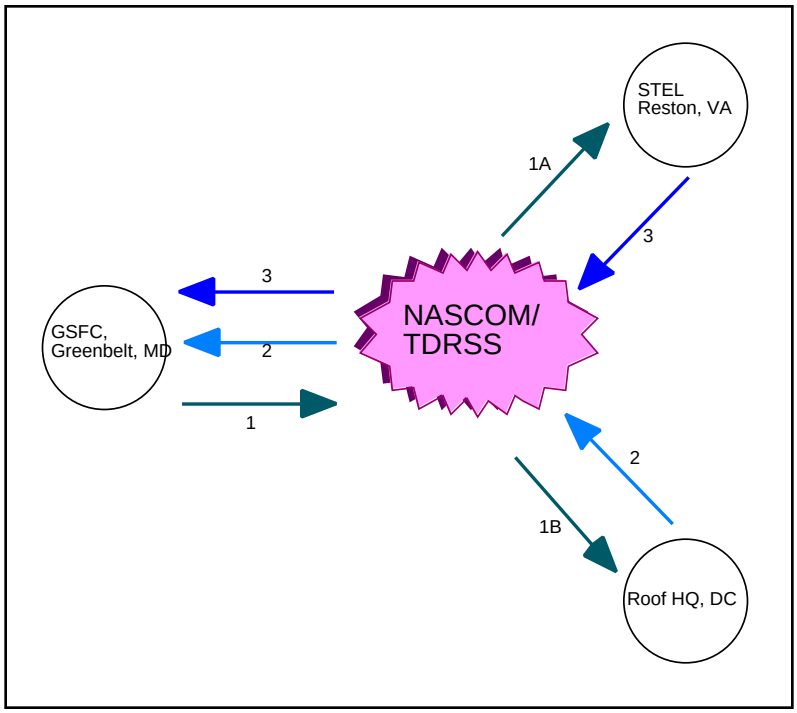

Figure 5: GLOBE System Demonstration

Conclusions

Software telemetry processing was successfully applied to the ground system for the GLOBE program. System costs were reduced in many areas due largely in part to a shortened design and implementation phase, commercial hardware to run software, and a fairly autonomous system requiring few man hours during operations. Also, maintenance costs were reduced, since in an "all software" solution, only standard vendor maintenance is required for the commercial hardware. No custom card maintenance is necessary. Thus, it makes sense to choose a software implementation of telemetry processing functions whenever performance requirements allow. The actual performance threshold will increase each year due to the advances in commercial CPU technology, and not due to the constant and costly hardware redesign that usually takes place inorder to increase performance. Clearly, this new approach possesses many possibilities in future low cost ground system applications.